ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DE SOFTWARE

Vehículo autónomo para gestión de trayectorias basado en Arduino y Raspberry

Autonomous vehicle for trayectory management based on Arduino and Raspberry

Realizado por
Francisco Antonio Gómez Guerrero
Tutorizado por
Luis Manuel Llopis Torres
Departamento
Lenguajes y ciencias de la computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, septiembre de 2016

Fecha defensa:
El Secretario del Tribunal

Resumen/Summary

Español

Este proyecto consiste en el desarrollo de un sistema de monitorización remoto de un robot autónomo.

El objetivo principal es hacer que el robot se desplace a través de un laberinto desde un punto de origen hasta un punto final de la forma más rápida posible.

El sistema cuenta con acceso a internet y una aplicación web accesible mediante cualquier navegador web en cualquier plataforma, ya sea en un smartphone, tablet u ordenador personal.

Mediante esa aplicación web el usuario podrá interactuar con el sistema, ya sea para conocer la situación y estado actual o para dar órdenes.

En este proyecto se utilizan tecnologías como HTML, PHP, Python, C y C# entre otras. Como plataformas hardware se utilizan Raspberry Pi y Arduino.

English

This project involves the development of a remote monitoring system for an autonomous robot.

The main goal is to make the robot move through a maze from an start point to an end point as quick as possible.

The system has internet access and an accessible web application through any web browser on any platform, whether is on a smartphone, tablet or personal computer.

Through this web application the user can interact with the system, either to know it’s current situation and state or to give orders.

In this project technologies such as HTML, PHP, Python, C and C # among others are used. Arduino and Raspberry Pi are used as hardware platforms.

Palabras clave/Keywords

Arduino, robot, laberinto, Raspberry, resolución, PHP, Python, C, multilenguaje, vehículo, autónomo, internet.

Maze, resolution, vehicle, autonomous, Arduino, robot, Raspberry, Python, C, multilanguaje , internet.

ÍNDICE

1.INTRODUCCIÓN

1.1 Introducción al proyecto.

Este proyecto pretende desarrollar un sistema autónomo capaz de desplazarse por el mundo físico desde un lugar de origen hasta un destino y que pueda ser controlado fácilmente por un usuario mediante una aplicación web.

Un robot autónomo es capaz de operar con un alto grado de autonomía, con un pequeño aporte de información proveída por un usuario. El robot presentado en este proyecto es capaz de desplazarse dentro de un laberinto desde un origen hasta un destino de la forma más rápida posible.

Durante el desarrollo del proyecto se utilizan tecnologías como PHP, Python, C y C# entre otros. Cada una de estas tecnologías es apropiada para un uso concreto, donde resultan de más utilidad, combinadas facilitan el desarrollo.

PHP se enfoca en presentar la interfaz al usuario por medio de la web. Python permite un código muy limpio, legible y sencillo, sirve de enlace para conectar las demás partes del sistema. Por último, C es un lenguaje muy potente para programar microcontroladores.

Las plataformas hardware elegidas en el proyecto son:

La Raspberry Pi es un ordenador de tamaño reducido, con potencia suficiente para este proyecto, será utilizado como soporte donde hacer funcionar PHP y Python. Ambas tecnologías cuentan con mucha información y soporte para funcionar sobre este ordenador.

Arduino es un microcontrolador, usado para facilitar el uso de la electrónica que será la que permitirá al robot desplazarse.

1.2 Objetivo del proyecto.

El objetivo principal del proyecto es que el robot sea capaz de llegar desde un punto de origen a otro de destino dentro de un laberinto.

Para ello el usuario solo debe suministrar un esquema de dicho laberinto que contendrá, la posición de partida, la posición objetivo y los lugares donde existan obstáculos que no pueden ser atravesados.

El sistema mediante una aplicación web permitirá al usuario:

- Consultar el estado actual del sistema.

- Si ya se le ha suministrado, permitirá ver el esquema del laberinto en el que se encuentra actualmente además de la posición actual del robot.

- Detener y reanudar la marcha del robot mientras se desplaza.

- Devolver el sistema al estado de partida.

2.TECNOLOGÍAS UTILIZADAS

Durante esta sección se expondrán todos los lenguajes de programación que han sido utilizados en la realización de este proyecto. Se incluirá una breve descripción del lenguaje y la funcionalidad que se ha conseguido con él.

2.1 PHP

Como forma de realizar contenido dinámico para el servidor web se ha utilizado PHP. Es un lenguaje de programación interpretado de uso general, ejecutado en el lado del servidor. Proviene del acrónimo en inglés de 'Pre Hypertext-Processor'. Se puede desplegar en la mayoría de servidores web y tiene una estructura parecida a las de Perl.

El funcionamiento básico de PHP es dotar de dinamismo a las páginas web, es capaz de recibir unos datos desde el cliente que se conecta a la página y mediante el intérprete de PHP procesarlos en el lado del servidor, siendo capaz de conectar a aplicaciones externas como bases de datos, servicios de almacenamiento, computación y finalmente generar una respuesta en formato HTML que es devuelta al cliente.

En este proyecto PHP formara el front-end del sistema, siendo la parte con la que el usuario interactúa directamente.

2.2 HTML

Siglas en inglés de 'HyperText Markup Language', es un lenguaje de marcado para páginas web y se usa para definir el contenido de la misma. Requiere un navegador web que lo interprete para representar su contenido.

En este proyecto HTML se usa para la página principal de la aplicación web y a su vez es el resultado de ejecución de otro código PHP.

2.3 Python

Es un lenguaje de programación interpretado de uso general. Su fuerte radica en la sencillez de su sintaxis y su rápida curva de aprendizaje. Soporta programación orientada a objetos, imperativa y funcional con tipado dinámico. Su interprete está disponible en multitud de dispositivos y arquitecturas por lo que es multiplataforma. Está licenciado bajo GNU.

En este proyecto un script para Python formará el núcleo del sistema recibiendo toda la información desde PHP procesándola y devolviendo una respuesta, manteniendo el estado más reciente del módulo arduino y comunicándose con este para transmitir instrucciones.

2.4 C#

Es un lenguaje de programación compilado. Fue estandarizado por Microsoft para su framework .NET. Tiene una sintaxis muy parecida a la de Java, aunque incorpora muchas mejoras derivadas de otros lenguajes como C y C++.

En este proyecto se ha usado C# para desarrollar un ejecutable que permite crear esquemas de laberintos y exportarlos al formato adecuado.

2.5 Socket

Es un concepto abstracto por el que dos programas pueden intercambiar un flujo de datos de manera fiable y ordenada, garantizando su integridad.

En este desarrollo el concepto se refiere a la interfaz de programación de aplicaciones (API) que permite la comunicación a través del protocolo TCP/IP de internet. Es necesario indicar un par de direcciones IP y puerto para la comunicación.

El API provee a menudo de métodos de alto nivel que permiten la creación, lectura y escritura del socket además de proporcionar mucha otra información como la dirección y puerto remotos de donde se recibe la conexión. Por último, provee de métodos para finalizar de forma correcta la comunicación.

Este API se ha usado en este proyecto en la programación con PHP y Python como método de comunicación.

2.6 Arduino(Lenguaje)

Arduino es un lenguaje de programación propio que se usa para programar el microcontrolador. El lenguaje está basado en Processing que es similar a C++. En concreto este lenguaje soporta todas las funciones estándar de C y algunas adicionales de C++.

El proyecto arduino en su totalidad tiene licencia de código abierto.

Cuenta con un gran número de funciones básicas para la entrada/salida tanto digital como analógica lo que facilita en gran medida leer información de los propios pines del microcontrolador. Además, la comunidad de Arduino posee un gran número de

librerías que implementan protocolos de comunicación con multitud de dispositivos electrónicos y sensores.

Arduino posee un entorno de programación propio, construido en Java lo que lo hace multiplataforma. Este entorno permite compilar y cargar código para multitud de microcontroladores de la familia arduino. Además de lo anterior también provee de una herramienta para la comunicación mediante el protocolo UART en la que se puede tanto leer los mensajes como enviarlos al arduino.

En este proyecto se ha usado el lenguaje Arduino y su entorno de desarrollo para construir el firmware que permite recibir información por el puerto serie, procesarla, almacenarla y mover los servomotores para hacer los movimientos necesarios.

2.7 PWM

Son las siglas en inglés de ‘pulse-width modulation’, modulación por ancho de pulso. Es una técnica en la que se modifica el ciclo de trabajo de una señal periódica para transmitir información o para controlar la cantidad de energía que se envía a una carga.

En este proyecto se usa para transmitir información a los servomotores controlados por arduino. Se genera una señal de onda cuadrada y se define el ciclo de trabajo que es relativo al tiempo que la onda es positiva frente al tiempo del ciclo total.

Mediante esta técnica se le indica al servomotor que posición tiene que adoptar.

2.8 Bootstrap

Es un framework de código abierto para diseñar sitios web. Contiene plantillas de diseño con tipografías, botones, formularios, etc.

Se ha usado para dar una mejor imagen en la interfaz de usuario web.

2.9 SSH

Son las siglas en ingles de ‘**S**ecure **Sh**ell’, se refiere a interprete de comandos seguro, es el nombre del protocolo y del programa que lo implementa. Permite conectar a la consola de comandos de un sistema remoto.

2.9 SFTP

Siglas en ingles de ‘SSH File Transfer Protocol’, protocolo seguro de transferencia de archivos. Permite la navegación por el sistema de ficheros de un

dispositivo remoto a través de una conexión SSH, la mayoría de programas con soporte para FTP (protocolo de transferencia de archivos) también lo tienen para SFTP.

3. SOFTWARE UTILIZADO

En esta sección se expondrán los programas que han sido utilizados para desarrollar este proyecto, una pequeña descripción y para qué se han usado.

Putty: Es un cliente de SSH para Windows desarrollado por Simon Tatham y de software libre. Ha sido utilizado durante el desarrollo desde Windows para conectar con el servicio de SSH de la Raspberry.

Este programa hace más sencillo interactuar con la Raspberry pues esta solo tiene que estar conectada a la red para acceder a su consola.

Filezilla: Filezilla es un cliente FTP multiplataforma de código abierto y software libre. Soporta los protocolos FTP, SFTP y FTP seguro.

En el desarrollo ha sido utilizado para enviar y recibir ficheros a la Raspberry desde Windows.

Sublime Text 2: Es un editor de texto y de código fuente desarrollado por Jon Skinner. Su característica adicional más útil es que cuenta con marcadores de sintaxis para muchos lenguajes.

Se ha usado para desarrollar los scripts de Python, las páginas de PHP y las páginas de HTML.

Visual Studio 2015: Es un entorno de desarrollo integrado para el sistema operativo Windows, propiedad de Microsoft. Soporta múltiples lenguajes de programación como C++, C#, Java, etc.

Se ha usado para desarrollar la interfaz en C# usando el asistente para la construcción de interfaces de usuario.

Fritzing: Es un programa de diseño electrónico libre. Permite crear esquemas de circuitos y exportarlos.

En este proyecto se ha usado para crear los esquemas para el montaje de arduino.

LucidChart: Es una herramienta de modelado UML online.

https://www.lucidchart.com

Ayuda en la creación de casi todo tipo de diagramas UML posibles, diagramas de clase, de casos de uso, diagramas de secuencia, etc.

Se ha usado para crear los diagramas de clase de este proyecto.

4. ANALISIS DE HARDWARE Y SOFTWARE

4.1 Requisitos hardware

Para el análisis hardware se expondrán que requisitos deberán cumplir los componentes integrados en el proyecto para que este tenga la capacidad hardware que se espera.

El robot debe contar con una capacidad de procesamiento que le permita que todas las tecnologías que usemos en él funcionen simultáneamente y de forma correcta. Debe contar con un sistema operativo y con conexión a internet.

El robot debe poder desplazarse por el espacio, para ello necesita, avanzar en línea recta y girar. Dado que el sistema es móvil debe ser compacto y ligero.

Por último, el sistema debe ser eléctricamente autónomo por lo que deberá llevar una fuente de energía.

4.2 Requisitos Software

Para el análisis software se describirán los requisitos que debe cumplir el sistema para tener la funcionalidad software que se espera.

El sistema debe permitir que múltiples usuarios interactúen con él de forma simultánea y desde múltiples plataformas. A todos ellos deberá proporcionarles una información actualizada sobre el estado actual en el que se encuentra, la posición dentro del laberinto si se conoce y el camino que ha recorrido.

Además de proporcionar información el sistema debe ser capaz de recibirla.

El sistema debe ser capaz de recibir del usuario ficheros con la información sobre el laberinto, ordenes de pausa y de reinicio.

El sistema debe ser estable ante los fallos, de haberlos, su propagación debe estar controlada. También debe ser modularizable para una posible expansión futura y proporcionar un registro de depuración.

El usuario debe contar con una herramienta que permita diseñar laberintos y exportarlos en forma de ficheros.

5. DISEÑO HARDWARE

Para cumplir con los requisitos expuestos en la fase de análisis se ha elegido el siguiente hardware para el proyecto.

5.1 Raspberry Pi

Es un ordenador de placa reducida de bajo coste desarrollado por la fundación Raspberry Pi.

Para este proyecto se ha elegido la Raspberry Pi 1 Modelo B, contienen una CPU con arquitectura ARM de la familia ARM11 mononúcleo a 700MHz con un conjunto de instrucciones RISC, 512 MB de memoria RAM compartida con la gráfica (se ha ajustado mediante software la memoria disponible para sistema y para gráfica siendo para esta última 16MB y el resto para el sistema).

La memoria de almacenamiento principal consiste en una tarjeta microSD de 16GB de capacidad marca Samsung conectada al puerto especifico de la Raspberry a través de un adaptador a SD.

La Raspberry tiene disponible muchos sistemas operativos, todos de código libre GNU/Linux como Debian (Raspbian), Fedora (Pidora), Arch (Arch ARM) entre otros. En este proyecto se ha usado Raspbian Jessie versión 4.4 disponible en la página oficial.

Posee dos puertos USB, en uno de ellos se conecta un adaptador wifi TL-WN723N dado que este modelo de Raspberry no posee conexión wifi propia, al otro USB se conecta un cable que pasa a USB tipo B conectado a la placa arduino.

El consumo medio de la Raspberry está alrededor de 700mA (3.5W) según dice en su especificación, quizás este consumo es un poco elevado dado que otro modelo puede funcionar con 500mA, pero solo se disponía del utilizado aquí. La alimentación proviene de un conector microUSB de 5 voltios.

Para anclar la placa Raspberry a la base del robot se ha usado una carcasa transparente con orificios para tornillos.

5.2 Arduino

Es una compañía de hardware libre que diseña y manufactura placas de desarrollo de hardware y software.

Para este proyecto se ha elegido la placa de circuito impreso Arduino Uno revisión 3 con un microcontrolador en su socket ATmega328. Este microcontrolador posee un oscilador interno que le permite trabajar a 8MHz, pero la placa de circuito incorpora otro oscilador externo que eleva la frecuencia hasta 16MHz. También posee 32KB de memoria flash, 2KB de memoria SRAM y 1KB de memoria EEPROM.

El microcontrolador posee 14 entradas y salidas digitales, 6 de ellas dan soporte para PWM y 2 se usan para la comunicación con el chip UART también incluido en la placa de circuito.

Este microcontrolador ofrece una facilidad adicional a la hora de cargar nuevos programas (sketches) en él. Tiene instalado un cargador (bootloader) que permite que el editor escriba la información de los programas de forma más rápida a través del puerto serie.

El consumo medio de la placa Arduino Uno es de unos 50mA aproximadamente. Recibe la alimentación por su entrada de USB tipo B desde la Raspberry por la que también recibe los datos del puerto serie.

5.3 Servomotores

Los servomotores son dispositivos parecidos a los motores de corriente continua, pero tienen la capacidad de ubicarse en cualquier posición dentro de su rango de operación y mantener dicha posición.

Los servomotores están compuestos de un motor, una caja reductora y un circuito de control.

Los servomotores hacen uso del PWM para controlar la dirección y la velocidad, arduino es capaz de mover estos motores en ángulos entre 0º y 180º.

Para este proyecto se ha elegido usar 4 servomotores modificados. El circuito de control está formado por un circuito integrado y un potenciómetro que gira solidario al movimiento del motor. Al girar el servomotor también gira el potenciómetro por lo que

el circuito sabe exactamente en qué posición se encuentra. La modificación realizada consiste en reemplazar ese potenciómetro por dos resistencias de valor 10Kohm para hacer creer al circuito que el potenciómetro siempre se encuentra en la posición central. De esa manera mediante software se indica que el servomotor vaya a su posición central para que se quede inmóvil y se le indica que vaya a alguna de sus posiciones en los extremos para que el motor quede girando continuamente en una dirección permitiendo el movimiento.

5.4 Potenciómetros

Es un elemento de electrónica, posee 3 terminales conectando los dos de los extremos a una tensión a regular se obtiene en el central una fracción de esa tensión dependiendo de la posición del potenciómetro.

En el proyecto se usan 4, uno por cada servomotor, para corregir el error introducido por las resistencias que se han usado para sustituir al potenciómetro. Por su tolerancia del 5% la posición central no queda bien establecido en el servomotor por lo que estos potenciómetros se ajustan manualmente y se corrige el error por software.

5.5 Batería

La alimentación usada en este proyecto es una batería externa para móviles, con una conexión USB a microUSB de salida, tiene 5000mAh de capacidad, salida de 5 voltios y puede ser recargada con cualquier cargador microUSB por su puerto de carga.

5.6 Protoboard y cableado

O placa de pruebas en español, es un tablero con orificios que se conectan eléctricamente entre sí de manera interna formando patrones en línea. Esta placa hace el montaje sencillo ahorrando soldaduras.

En el proyecto se usa para conectar los servomotores a la alimentación y para alojar los potenciómetros.

Se usan cables con punta metálica adecuados para la placa de protoboard y los conectores de arduino.

6 MONTAJE

El robot está montado sobre una lámina de PVC de 18cmx18cm que sirve como base.

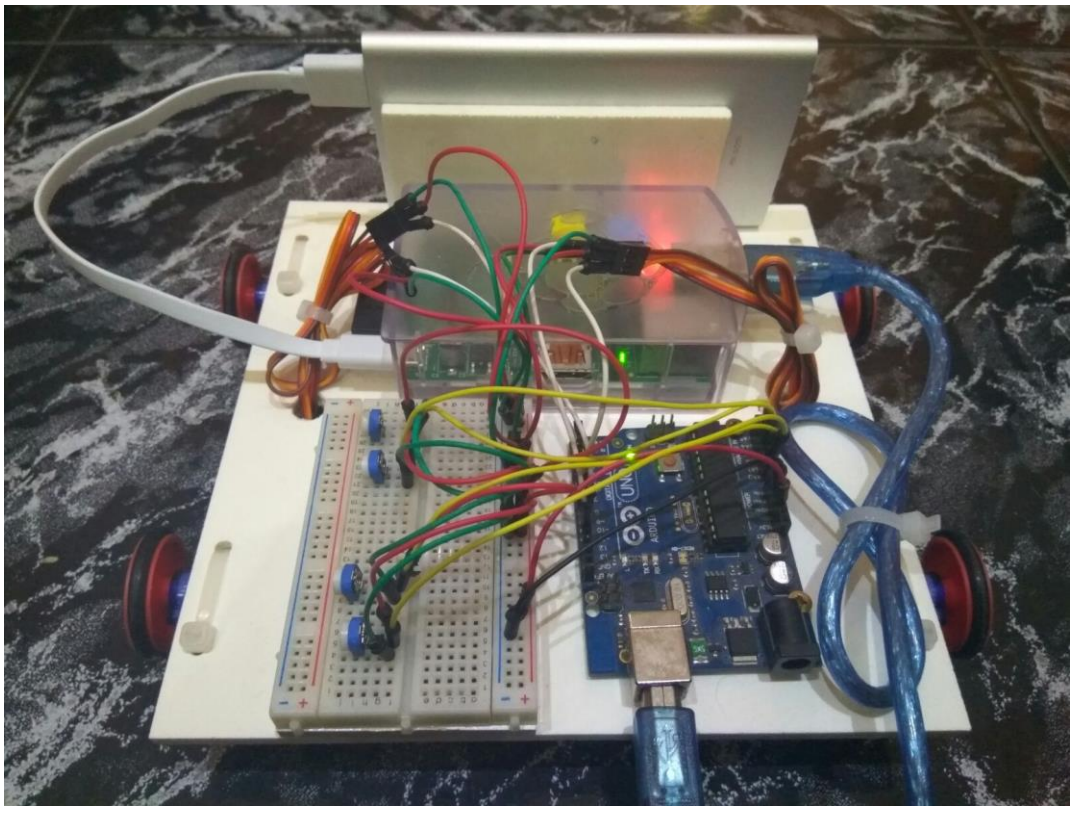
En la parte trasera se han unido con pegamento de PVC dos laminas verticales del mismo material para sostener la batería.

Pegada a la parte trasera y hasta el centro se encuentra la Raspberry metida dentro de su carcasa. Esta se ha unido a la base con dos tornillos de 1cm de longitud.

En la parte frontal se han colocado la placa protoboard y el arduino, la placa protoboard está fijada con cinta de doble cara a la base y el arduino no dispone de funda y va fijado a la base con dos tornillos de 1cm directamente en la placa.

Los servomotores están unidos cada uno a la base mediante bridas que pasan por dos agujeros. Están dispuestos formando un cuadrado.

Foto del resultado hardware final.



En cuanto a las conexiones de cables, el arduino va unido a la Raspberry mediante un cable USB tipo B (Azul en la foto).

Todos los servomotores toman su alimentación de las líneas de positivo (cables rojos) y negativo (cables verdes) de la placa protoboard que a su vez son alimentadas por la salida de 5v (cables rojos) de arduino y GND (cables negros) respectivamente. Los cables de salida de la señal PWM para los servomotores se toman de los pines del arduino numerados como:

- Motor delantero derecha pin 6.

- Motor delantero izquierda pin 9.

- Motor trasero derecha pin 10.

- Motor trasero izquierda pin 11.

Los potenciómetros también toman su alimentación de las líneas de la protoboard (rojos y verdes) y su pin central va a las entradas analógicas del arduino nombrados como:

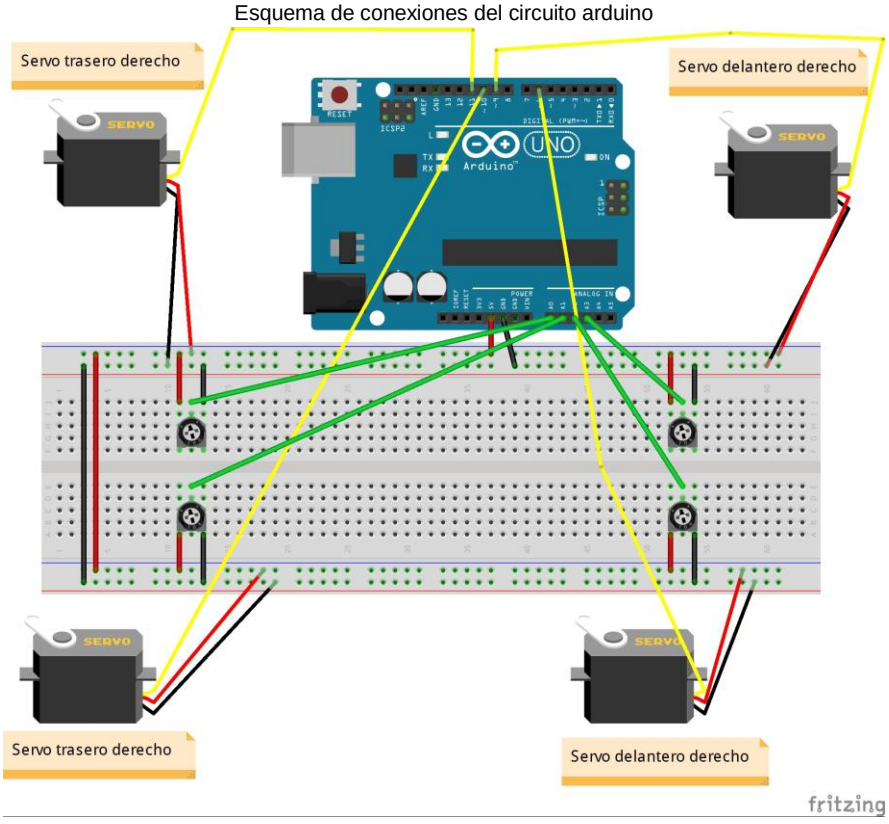
- Potenciómetro motor delantero derecha pin A1

- Potenciómetro motor delantero izquierda pin A0

- Potenciómetro motor trasero derecha pin A3

- Potenciómetro motor trasero izquierda pin A2

Esquema de conexiones del circuito arduino



7. DISEÑO SOFTWARE

En esta sección se expondrá la solución software a la que se ha llegado y que pretende cubrir todos los requisitos generados durante la fase de análisis.

El sistema está dividido en tres capas muy apreciables modelo, vista y controlador. PHP generará la vista para el usuario, Python formará el controlador y el módulo de arduino formará el modelo.

La herramienta para diseñar y exportar los esquemas de laberintos se desarrollará en C#.

Se expondrá el sistema operativo elegido y cada parte que compone el software como módulos separados.

La forma en la que estas tecnologías se comunican se abordará más adelante.

7.1 Sistema operativo

El sistema operativo seleccionado ha sido Raspbian Jessie.

Un sistema operativo GNU/Linux de código libre basado en Debian. Disponible en la página oficial de Raspberry, es compatible con todos los modelos de Raspberry Pi.

En este momento la última versión estable del sistema es la 4.4 publicada el 21/05/2016.

Cuenta con un intérprete de Python preinstalado además gracias a su sistema de paquetes se podrá instalar de forma sencilla el servidor web apache con soporte para PHP.

7.2 Módulo PHP

En primer lugar, se abordan los requisitos referentes a la interfaz.

El diseño se ha dividido en 5 ficheros, donde cada uno hace una función propia.

Fichero ‘index.html’, es un fichero en formato HTML, la función de este fichero es básica, presenta toda la información junta y de manera estructurada en una misma aplicación web. Incorpora los 3 botones con los que el usuario podrá interaccionar con el sistema además de dos iframe que referencian a otros dos ficheros ‘mapa.php’ y ‘estado.php’ que muestran información.

Fichero ‘operaciones.php’, es un fichero en formato PHP, su función es recibir el formulario generado por el navegador cuando un usuario presiona un botón en la página de ‘index.html’. Analiza la información contenida en el formulario y envía un comando acorde al botón pulsado al módulo Python además de más información relevante.

Fichero ‘mapa.php’, es un fichero en formato PHP, contiene una línea de metadatos en HTML para refrescar la página en un intervalo de tiempo. Su función principal es conectar con el módulo Python a través de un socket y solicitar la información del mapa actual si lo hubiese, una vez obtenida la información genera una salida HTML con una tabla que representa de forma visual al usuario el laberinto.

Fichero ‘estado.php’, es un fichero en formato PHP, contiene la misma línea de metadatos del fichero ‘mapa.php’. Su función principal es conectar con el módulo Python a través de un socket y solicitar la información del estado actual del sistema, una vez obtenida la información genera una salida HTML que depende del estado recibido.

Fichero ‘conexion.php’, es un fichero en formato PHP, contiene una función para crear el socket de conexión que usan el resto de ficheros.

7.3 Módulo Python

El módulo de Python hace la función de controlador dentro del patrón MVC.

El desarrollo en este lenguaje se ha dividido en 5 scripts.

Script ‘Principal.py’, como su nombre indica es en el que se da comienzo al programa. Su función principal es establecer los parámetros de control inicial.

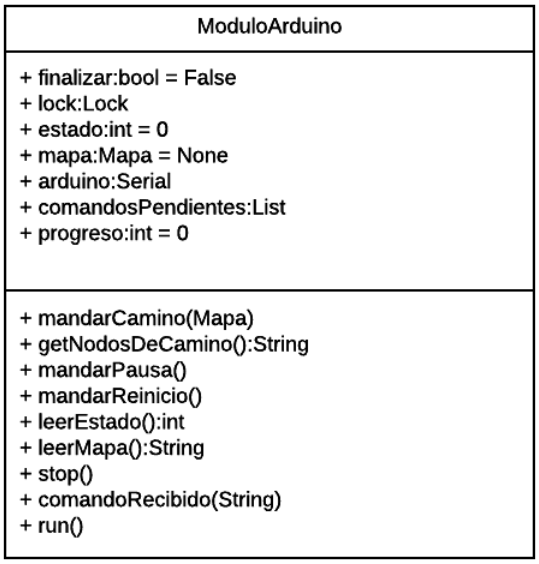
Al comienzo del script se analizan los parámetros pasados al ser iniciado desde la línea de comandos. Estos parámetros pueden ser ‘PHP’, ‘arduino’ y ‘conexión’. Si el script es llamado con alguno o varios de estos parámetros se mostrará la información de depuración correspondiente.

Crea la instancia de la clase ModuloArduino y después abre un socket de red para escuchar en la interfaz y puerto en la que va a trabajar. Cuando llega una conexión nueva se crea una instancia de la clase ModuloPHP que atiende la conexión. La señal

elegida para finalizar el programa es la interrupción por teclado, cuando esta señal llega se finalizan de forma segura todos los hilos creados.

Script 'ModuloArduino.py', contiene la clase ModuloArduino, esta clase es la encargada de la comunicación con arduino, extiende de la clase Thread, contenida en la biblioteca Threading, por lo que opera de forma separada en su propio hilo.

La comunicación con la placa arduino se hace mediante la clase serial, contenida dentro de la biblioteca 'Serial' de Python.



Esta clase mantiene datos sobre el estado de la placa arduino, estos datos son actualizados normalmente mediante comandos que son recibidos desde la misma.

Para satisfacer el requisito de conexiones simultaneas de varios usuarios se ha creado una lista de comandos pendientes, cuando una clase externa usa los métodos de esta clase para mandar comandos a arduino estos se añaden a dicha lista. Para controlar el uso compartido de esta lista entre varios hilos se usa un mecanismo de protección conocido como Lock, proveído por la biblioteca Threading, ese es el propósito de la variable lock. El Lock se cierra cada vez que se hace uso de la lista y se abre cuando se finaliza.

La función más relevante es run(), que se inicia cuando se crea el hilo. La función comprueba periódicamente si hay algún comando pendiente en la lista. Si lo hay coge el primero y lo manda la placa arduino. Después la función comprueba si hay algún comando enviado desde la placa arduino, si lo hay lo lee y lo envía a la función comandoRecibido(), que lo procesa.

Se ha elegido el periodo de medio segundo para el envío de datos dado que el buffer de entrada de la placa arduino es muy pequeño, de esta manera se asegura que no se pierdan datos en el envío, proporcionándole a la placa arduino suficiente tiempo para leer el buffer y limpiarlo.

Para controlar el periodo de espera se usa la función 'time.sleep', que detiene la ejecución del hilo durante el tiempo indicado, mientras que el hilo está detenido no se pueden capturar las excepciones, por ejemplo la que hace finalizar al programa. Por ello se cuenta con la variable booleana 'finalizar' que establecida a 'True' indica que en la siguiente iteración dentro de run() finalizará el hilo de forma segura.

El resto de funciones no tienen un comportamiento muy complejo.

-mandarCamino(mapa): Recibe como parámetro una instancia de la clase mapa, actualiza el estado del objeto tipo Mapa asociado al laberinto actual y encola el comando para enviarlo.

-getNodosDeCamino(): Devuelve un String que contiene un array continuo separado por espacios con los VALORES DE CAMINO actuales para cada posición del laberinto.

-mandarPausa(): Encola en la lista de comandos pendientes el comando de pausa.

-mandarReinicio(): Encola en la lista de comandos pendientes el comando de reinicio.

-leerEstado(): Devuelve un entero que representa el estado del dispositivo.

-leerMapa(): Devuelve el objeto de tipo Mapa asociado al laberinto actual.

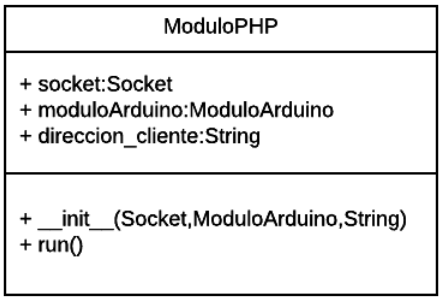
-stop(): Establece la variable 'finalizar' a 'True' para finalizar el hilo lo más pronto posible.

-comandoRecibido(String): Procesa las líneas de comando recibidas desde arduino.

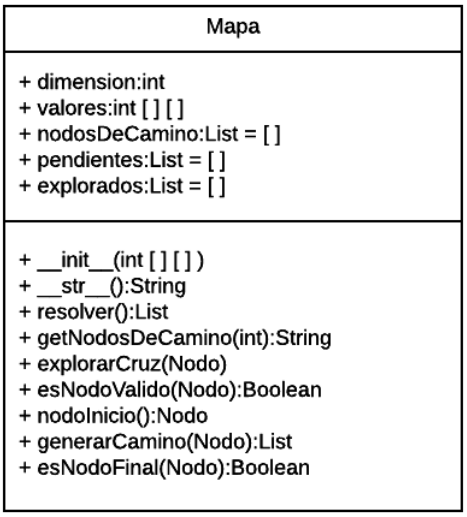
Script 'ModuloPHP.py', contiene la clase 'ModuloPHP', esta clase se ocupa de atender las conexiones entrantes desde el módulo de PHP. Extiende a Thread por lo que opera de forma separada en su propio hilo. Existen tantas instancias de esta clase como conexiones en ese momento.

En el momento de su creación esta clase recibe el socket que se creó para la conexión que atendería y el módulo arduino sobre el que va a trabajar.

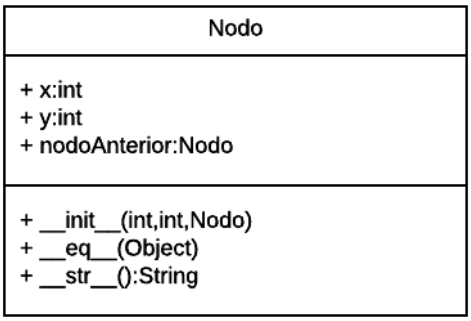
Su función principal es run(), que lee desde el socket el comando enviado por PHP, lo analiza, ejecuta y responde según sea necesario. Después de atender la conexión el hilo finaliza.



Script 'Mapa.py', contiene la clase 'Mapa', esta clase estructura los datos para representar el mapa de un laberinto. Además, contiene métodos útiles para resolver dicho laberinto y otra información adicional.



La clase mapa también contiene una clase auxiliar anidada llamada Nodo. Se usa para almacenar la información referente a una posición en el mapa, accesible desde coordenadas (x,y). También contiene una referencia si la hubiese al nodo anterior a este.



La variable valores es un array doble (Matriz) que contiene los VALORES DE MAPA asociados a cada posición (x,y) dentro de la matriz.

La función destacable de la clase Mapa es resolver(), esta clase devuelve una lista con los VALORES DE DIRECCION en las que se tiene que avanzar para llegar desde la casilla de inicio hasta la final.

Para resolver el laberinto se usa el **algoritmo A*** que nos asegura que de haber un camino siempre se encontrará y además será óptimo.

El algoritmo funciona de la siguiente manera. Se busca el nodo inicial (coordenadas x e y donde se comienza) y se añade a la lista pendientes y explorados. Ahora de forma recursiva se comprueba si hay algún nodo dentro de la lista pendientes, si no hay se acaba el proceso pues el laberinto no tiene solución. Si hay algún elemento se comprueba si cumple las condiciones para ser nodo final mediante la función esNodoFinal(),si lo es se pasa dicho nodo a la función generarCamino() que devuelve el camino final. Si no es el nodo final se utiliza la función explorarCruz() que genera nuevos nodos y los agrega a la lista de pendientes.

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

El resto de funciones son auxiliares.

-getNodosDeCamino(int): Esta función devuelve un String con la matriz del mapa en forma de un array continuo separado por espacios, los valores corresponden al VALOR DE CAMINO en esa posición. El parámetro que recibe es el progreso y sirve para indicar en que parte del recorrido se encuentra el vehículo en ese momento para poder devolver el VALOR DE CAMINO adecuado.

-explorarCruz(Nodo): Esta función genera los 4 nodos adyacentes al que se le pasa como argumento. Comprueba si los nodos generados son válidos mediante la función esNodoValido(Nodo) y si no se encuentran ya dentro de la lista de explorados, de cumplir estas dos condiciones los nuevos nodos se añaden a la lista de pendientes y a la de explorados.

-esNodoValido(Nodo): Esta función comprueba si un nodo es válido. Las condiciones para ser válido son, que no esté fuera de los límites de la matriz y que ese nodo sea transitable según el VALOR DE MAPA de esa posición.

-nodoInicio(): Esta función devuelve, si existe, el nodo de comienzo del mapa que será el que contenga el VALOR DE MAPA usado para indicar el comienzo.

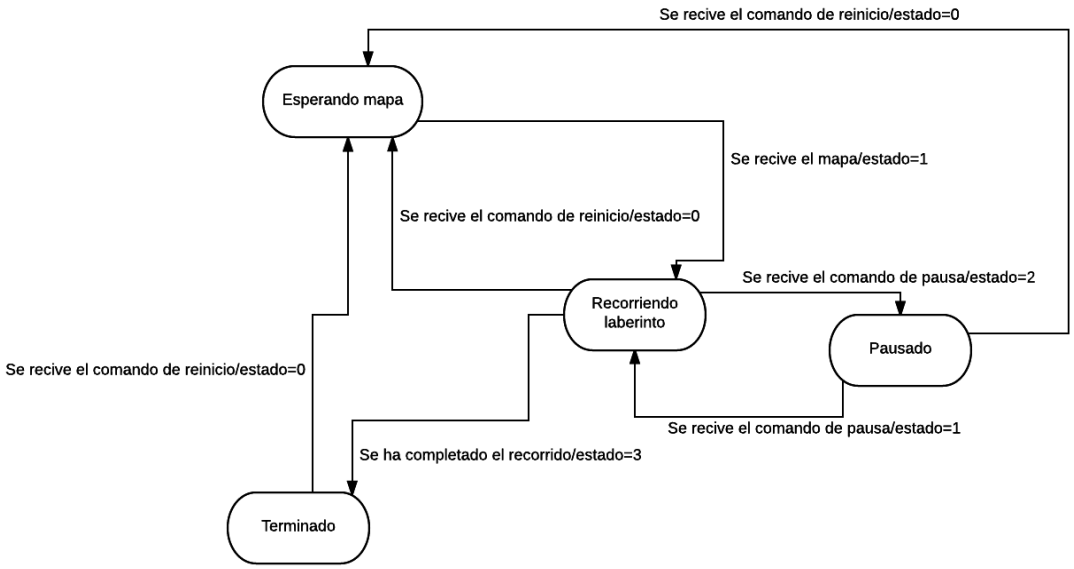
-generarCamino(Nodo): Esta función devuelve el camino a seguir en forma de VALORES DE DIRECCION. Recibe como parámetro el nodo final encontrado en otra fase y reconstruye el camino hacia atrás siguiendo la cadena de predecesores mediante la variable nodoAnterior asociada a cada nodo, hasta que alcance un nodo en el que no exista predecesor.

-esNodoFinal(): Esta función comprueba si el nodo que recibe como parámetro es el nodo final del laberinto, es decir si el VALOR DE MAPA de esa posición coincide con el final.

Script 'settings.py', contiene variables globales compartidas por todos los otros scripts. Estas variables son referentes al nivel de información que el script proporcionará por línea de comandos, sobre los eventos que están ocurriendo.

7.4 Módulo Arduino

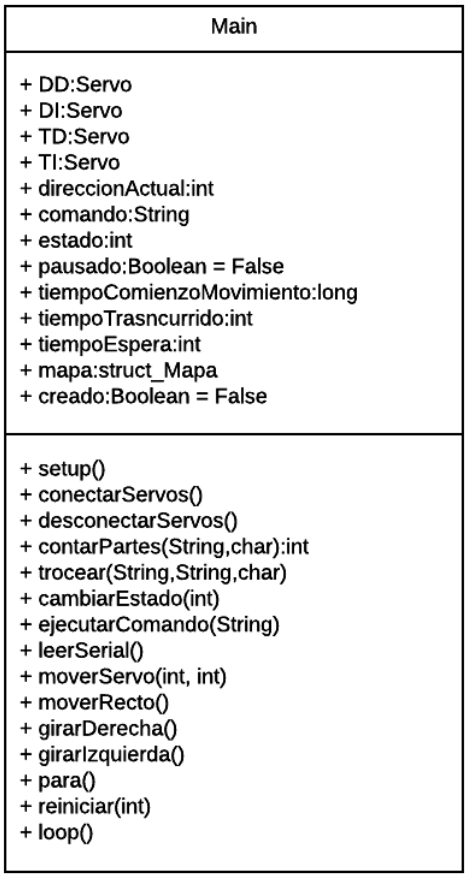
El módulo arduino corresponde al modelo dentro del patrón MVC. El desarrollo consistirá en un programa o sketch que será gravado en la placa arduino, es decir se está programando su firmware. Ha sido diseñado como una máquina de estados.



El programa comienza en la función setup() en la que se inicia la comunicación serie y se ponen los motores en reposo.

Después de la ejecución setup() se ejecuta en bucle la función loop().

En cada vuelta del bucle se ejecutan dos tareas, leer desde el puerto serial y realizar el comportamiento según él estado actual.



La función leerSerial() se ejecuta dentro de loop() y se encarga de lo primero, lee desde el puerto serie byte a byte transformándolo en un carácter ASCII y almacenándolo en la variable 'comando' hasta que encuentra el carácter de finalización. Una vez encontrado llama a la función ejecutarComando(String) con la línea completa del comando.

Una vez leído el puerto serie se usa un bloque switch para modelar el comportamiento de máquina de estado.

En los estados 'Esperando mapa', 'Terminado' y 'Pausado' no se realiza ninguna función, solo se mantiene el dispositivo dormido con la función delay() durante 100ms antes de continuar con el bucle. Se hizo de esta forma por la facilidad para ampliar funcionalidades si fuese necesario.

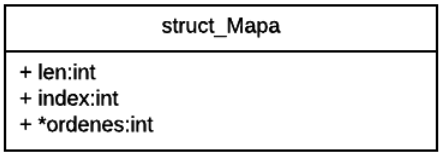
Durante el estado 'Recorriendo laberinto' hay un comportamiento activo, una vez el sistema se encuentra en este estado ya están disponibles en la variable 'mapa' la información sobre los movimientos que se deben realizar.

Para controlar el tiempo que los servomotores deben estar en movimiento se toma cada orden de movimiento de una en una. Cuando se ejecuta una acción, ya sea girar o avanzar, la función encargada de dicho movimiento establece un tiempo de espera en la variable 'tiempoEspera' que indica el tiempo que tendrá que pasar para ejecutar la siguiente orden, en todo ese tiempo los servomotores estarán ejecutando dicha orden.

Dependiendo del valor de 'direccionActual' que indica a qué dirección está apuntando el robot en ese momento se decide la acción a realizar, si se está apuntando a la misma dirección en la que se tiene que movernos solo se avanza hacia delante con la función moverRecto() y se incrementa el valor de índex en el mapa. Si la dirección no concuerda se usa girarIzquierda() para rotar el robot hacia la izquierda, o bien girarDerecha() para rotarlo hacia la derecha o cambiar de sentido.

La variable 'mapa' es una instancia de la estructura Mapa.

-len: Almacena la longitud del array de direcciones.



-index: Almacena la posición del array en la que se encuentra actualmente. También vale para controlar el progreso.

-*ordenes: Es un array que contiene todos los VALORES DE DIRECCION que componen el laberinto.

Este array se crea y destruye dinámicamente cada vez que se recibe un mapa nuevo.

El problema surgido durante la modificación de los servomotores, en el cual estos no estaban bien centrados, deriva en que los servomotores o bien están continuamente girando en una dirección o están temblando.

El valor central de un servomotor, donde no debería moverse es 90. Algunos tienen su centro más bajo o más alto.

-moverServo(int,int): Se le proporciona a la función el servo que se debe mover y la velocidad a la que debe hacerlo, una velocidad negativa hará que el servo gire en dirección opuesta.

Esta función calcula el ángulo al que debe establecer cada servo mediante una operación matemática que depende de los valores leídos del potenciómetro asociado a cada servo. Este valor puede estar comprendido entre [0,1024] que da la lectura analógica de arduino y se escala en el rango [0,30] para después sumarle a una cantidad fija.

Angulo = 75º + valor_escalado +/- velocidad

El signo de la velocidad viene dado por la posición de los servomotores, puesto que unos están girados con respecto a los otros, los servos colocados a la derecha poseen un signo y los colocados a la izquierda el otro.

-conectarServos() y desconectarServos(): Estas funciones son usadas para evitar que los servomotores tiemblen durante el tiempo que están sin usarse y ahorrar batería.

-para(): Usa la función moverServos(int, int) estableciendo la velocidad a 0.

-contarPartes(String,char): Recibe una cadena y un delimitador **,** devuelve cuantas veces se encuentra este delimitador en la cadena y le suma 1.

-trocear(String[],String,char): Se le proporciona un array, una cadena de texto y un delimitador. Divide la cadena en partes por el delimitador y coloca cada parte en el array.

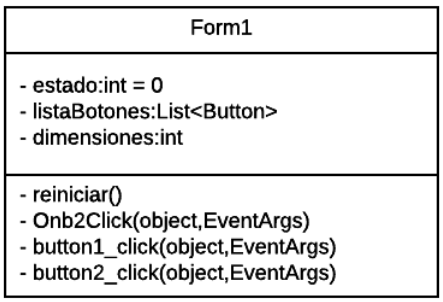
-cambiarEstado(int): Hace la transición de estado en la máquina de estados, transmite el nuevo estado al módulo Python y activa o desactiva los servomotores.

-ejecutarComando(String): Recibe una línea de comando completa y la ejecuta.

-reiniciar(int): Reestablece todas las variables y cambia el estado al recibido por parámetro.

7.5 C#

La variable estado indica que tipo de casilla se está colocando, con el valor 0 la primera casilla seleccionada será el origen.



Cuando se identifica pasa al valor 1 en el que se indica el final.

Cuando el final también es establecido pasa a valor 2 y se identifican las casillas transitables.

-reiniciar(): Devuelve todas las variables a su estado de inicio.

-Onb2Click(object,EventArgs): Este método es llamado cuando se pulsa un botón de la cuadricula del mapa y cambia el valor de ese botón según el estado actual.

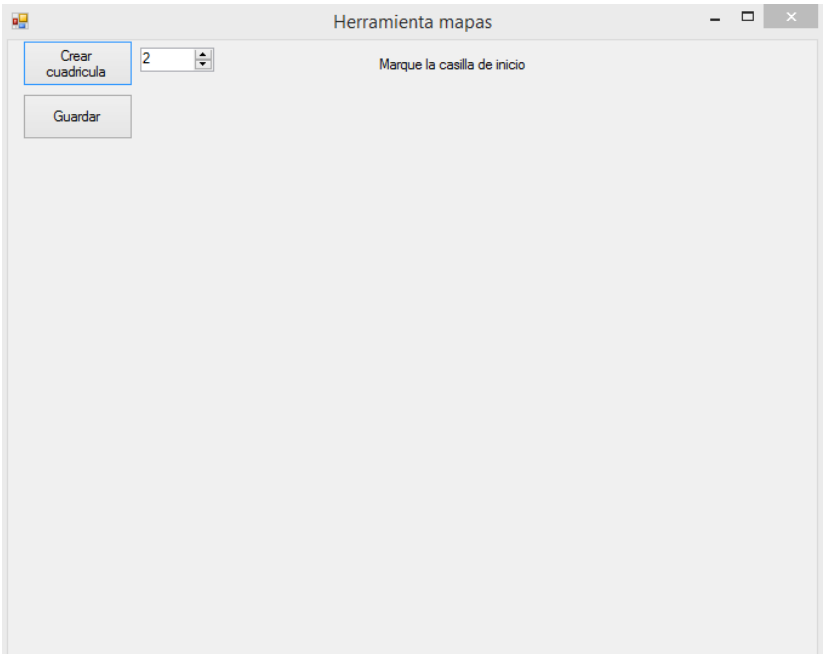
-button1_click(object,EventArgs): Es el método llamado por el botón 'Crear cuadricula', genera una matriz de botones en la parte central de la interfaz, la matriz tiene las dimensiones que el usuario ha indicado en un control numérico.

-button1_click(object,EventArgs): Es el método llamado por el botón 'Guardar', abre una ventana de dialogo para seleccionar el nombre del fichero y su ubicación, una vez seleccionado guarda en ese fichero el mapa.

El formato utilizado para almacenar el mapa es el siguiente.

N n_0 n_1 … n_{N^2-1}

N identifica las dimensiones y n_0 n_1 … identifican los VALORES DE MAPA.


Herramienta mapas
Crear cuadricula
2
Guardar
Marque la casilla de inicio

8. PRUEBAS

Durante el desarrollo y una vez finalizado se han elaborado una serie de pruebas para comprobar el correcto funcionamiento del sistema.

-Pruebas de carga.

El sistema es accesible mediante web, por lo que muchos usuarios pueden conectarse al mismo tiempo. Se han realizado pruebas con 30 conexiones web simultaneas. No se ha detectado ninguna anomalía, la conexión con Python toma un tiempo muy breve y la configuración recomendada para Apache establece el límite de conexiones en 100 por lo que será este quien limite el número de usuarios simultáneos.

-Pruebas de autonomía.

El consumo medio en reposo se ha medido entorno a los 800mA que se eleva a 1,3A cuando todos los servomotores están activos. Dado que la batería tiene una capacidad de 5000mAh y una eficiencia media del 75% contamos con 3750mAh de capacidad real.

Si dividimos la capacidad real entre el consumo medio en reposo obtenemos

3750mAh/800mA = 4.68 horas de funcionamiento en reposo. En la práctica la batería a plena carga pudo mantener el sistema en funcionamiento más de 5 horas ya que el cálculo ha sido muy conservador y no se dejó que la batería llegara a su mínima carga por seguridad para el sistema.

Si repetimos el mismo cálculo para el consumo a plena carga obtenemos:

3750mAh/1300mA = 2.88 horas de funcionamiento a plena carga. Este resultado no ha sido verificado en la práctica.

-Prueba de precisión en el desplazamiento recorrido.

Se ha sometido al sistema a pruebas relacionadas con la precisión cuando se desplaza.

No se ha podido tener en cuenta durante el desarrollo la imprecisión que generan 4 motores girando al mismo tiempo, dado que en la práctica no todos funcionan a la misma velocidad, no todas las ruedas tienen la misma adherencia y no todas las superficies son iguales.

Por lo tanto, se ha medido una variación de la trayectoria cuando el vehículo se desplaza en línea recta de unos 10 cm sobre la posición correcta.

Cuando el vehículo gira sobre sí mismo se produce un error de unos 10º respecto al ángulo correcto.

Todas estas imprecisiones se suman con cada movimiento del vehículo, si el recorrido es muy largo la posición final puede diferir mucho de la esperada.

9. INTEGRACIÓN

Una vez finalizado el desarrollo del software hay que implantarlo en el sistema real.

Para preparar la placa Raspberry pi se hay que instalar RASPBIAN JESSIE descargado desde la página oficial de la fundación Raspberry pi.

https://www.raspberrypi.org/downloads/raspbian/

Se descarga la imagen del sistema, en este proyecto este proceso se ha realizado en Windows. Con el programa win32diskimager se ha procedido a grabar en la tarjeta microSD usada como memoria de la Raspberry la imagen de Raspbian.

Una vez grabada se ha conectado a la Raspberry la tarjeta, un monitor, un ratón y teclado y el adaptador wifi.

En el primer inicio el sistema nos da una serie de opciones para la configuración inicial.

Los cambios que se han realizado han sido:

-Expandir el sistema de ficheros, para que se utilice toda la capacidad de la memoria.

-Se ha establecido la frecuencia de trabajo a la normal 700Mhz.

-Se ha reducido la RAM de la GPU a 16MB y se ha aumentado la RAM de sistema a 496MB

-Se ha habilitado el soporte para SSH.

-Se ha seleccionado como pantalla de inicio predeterminada el modo consola, donde se pide usuario y contraseña.

Una vez finalizados el sistema se reiniciará. Cuando vuelva a iniciar, introducimos las credenciales por defecto (Usuario: pi, Contraseña: raspberry) y procedemos a configurar la conexión wifi manualmente.

Se introduce el comando:

‘sudo nano /etc/network/interfaces’

Se abrirá el editor nano para el fichero interfaces. La configuración exacta depende del router que dé servicio a dicha conexión wifi. En este proyecto el fichero debe quedar así:

```
auto eth0
allow-hotplug wlan0

iface lo inet loopback

iface eth0 inet manual

iface wlan0 inet static
        address 192.168.1.133
        netmask 255.255.255.0
        network 192.168.1.0
        gateway 192.168.1.1
        wpa-passphrase aaaaaaaa
        wpa-ssid tfg
```

Presionamos CTRL+X y presionamos ‘y’ para guardar los cambios.

Después de este paso se vuelve a trabajar desde Windows, accediendo a la consola mediante SSH usando Putty como cliente usando la dirección de red que le asignamos anteriormente.

Una vez iniciemos sesión será necesario ejecutar estos comandos consecutivamente para actualizar las librerías y binarios del sistema a la última versión.

```
sudo apt-get update
sudo apt-get grade
```

Raspbian ya cuenta con un intérprete de Python por lo que no será necesario instalar nada.

Para instalar el servidor apache se introduce en la consola el siguiente comando.

```
sudo apt-get install apache2 -y
```

Y para instalar el soporte para PHP

```
sudo apt-get install php5 libapache2-mod-php5 -y
```

Para la instalación de Bootstrap se ha descargado desde su página oficial un fichero .zip que contiene las carpetas css, fonts y js.

http://getbootstrap.com/getting-started/

Estas carpetas deberán se ubicadas en '/var/www/html' directorio por defecto de apache.

Por último, instalamos el programa 'screen', este programa permite que una consola local pueda quedar abierta al finalizar una conexión remota. Se usará para iniciar el script de Python en cada inicio y poder retomar la consola local para ver la salida desde una consola remota.

```
sudo apt-get install screen
```

El sistema ya cuenta con todos los programas necesarios para funcionar, solo resta colocar los ficheros de nuestro programa en sus respectivos lugares.

Los correspondientes a PHP se colocan en '/var/www/html'

Estos ficheros son conexión.php, estado.php, mapa.php y operacion.php

Los ficheros correspondientes a Python pueden colocarse en cualquier lugar, en este proyecto se han ubicado en '/home/pi/Desktop/python'. Los ficheros son Mapa.py, ModuloPHP.py, ModuloArduino.py, settings.py y Principal.py.

Como paso final debemos editar el fichero de inicio de sistema '/etc/rc.local' añadiendo esta línea.

```
screen –d –m python /home/pi/Desktop/Python/Principal.py
```

Para que el script de Python se ejecute en cada inicio.

10. INSTRUCCIONES DE USO

El proceso comienza realizando un esquema del laberinto mediante una interfaz gráfica para Windows, se le indica las dimensiones del mapa que será cuadrado y está dividido en casillas y se pulsa el botón ‘Crear cuadricula’, mediante clicks de ratón se marca la casilla de inicio y la de finalización, por defecto el resto de casillas son paredes y el robot no puede caminar por ellas, para hacer caminos se marcan las casillas que sí son transitables.

Una vez terminado el esquema se pulsa el botón ‘Guardar’ y se genera un archivo de salida que contiene toda la información.

Para la interacción del usuario con el robot se usa una aplicación web accesible a través de un navegador web que contiene 3 botones y dos campos de información. Un campo muestra el estado actual de todo el sistema y el otro si procede muestra el mapa que ha sido proporcionado anteriormente y la posición actual del robot en el mismo. Continuando con el proceso en la aplicación web se nos pide que seleccionemos desde nuestra computadora el esquema realizado anteriormente, una vez seleccionado, el botón ‘Enviar’ subirá el archivo a la aplicación. Después de resolver el laberinto el robot comenzara a desplazarse por él hasta que alcance la posición final. Los dos botones restantes de la aplicación son ‘Pausar’ que detiene momentáneamente el recorrido hasta que vuelva a pulsarse y ‘Reiniciar’ que devuelve todo el sistema al estado de inicio.

11. CONCLUSIONES

Este proyecto me pareció mucho más complejo en el momento en el que lo plantee con mi tutor.

He usado varias tecnologías que no conocía previamente como Python y PHP ambas con tipado dinámico, algo que no había visto antes. Por suerte ya tenía bastante experiencia con Arduino, plataforma con la que siempre he querido realizar un proyecto y que me motivó a la hora de elegir este.

Durante todo el grado no me había encontrado con la necesidad de comunicar varias partes de un sistema complejo que estén realizadas en lenguajes distintos y al comienzo requirió mucho análisis de las posibilidades que me ofrecía cada uno de esos lenguajes.

A medida que iba avanzando en el desarrollo se hacía más fácil y rápido pues ya conocía la sintaxis de cada lenguaje y por la grandísima cantidad de información disponible en internet, en páginas web especializadas y foros de usuarios con dudas similares.

También han surgido muchos problemas inesperados durante la fase de implementación lo que me ha enseñado que la de diseño requiere un tiempo similar o mayor. Así un proyecto con un diseño muy estudiado y pulido da como resultado una implementación más rápida, clara y sencilla.

Realizar el proyecto por partes ha ayudado mucho a contener los errores, controlando y analizando los mensajes que recibe cada módulo me aseguro que los errores no se propagan. Esto hace el sistema más estable y lo hace mucho más sencillo de depurar.

Desde el primer momento el tema de desplazar un vehículo físico por el mundo real ya me pareció una tortura, del mundo del software donde todo es en gran medida predecible di el salto al hardware, donde controlar las variables de entorno se vuelve imposible, cosa que ha quedado demostrada cuando he hecho pruebas y he observado que conseguir que el vehículo se desplace en una línea recta era muy complicado.

12. PROTOCOLOS DE COMUNICACIÓN

Para comunicar todas las tecnologías se han usado comandos, es decir, líneas de texto con parámetros variables separados por un delimitador y con un carácter de final de línea para delimitar cada línea.

En todos los protocolos se ha usado el espacio como carácter delimitador y el carácter '\n' (final de línea) como carácter de finalización.

12.1 Protocolo de comunicación PHP-Python

La comunicación entre PHP y Python se realiza de forma síncrona, en la misma conexión PHP transmite el comando hacia Python y si así se requiere se manda la información de vuelta, después se cierra la conexión. La conexión es creada con sockets, Python escucha en el puerto 9999 de la interfaz localhost, esa misma dirección es usada por PHP para conectar.

Comando: 'exit'

Formato: exit

Descripción: Cierra la conexión, no se usa en la solución final, pero fue útil durante la fase de pruebas y desarrollo.

Formato salida: -

Comando: 'getstado'

Formato: getstado

Descripción: Devuelve el VALOR DE ESTADO del módulo Arduino.

Formato salida: Devuelve el VALOR DE ESTADO que identifica al estado.

Comando: 'getmapa'

Formato: getmapa

Descripción: Devuelve el VALOR DE MAPA de todas las posiciones del mapa en un array continuo. Primero N identifica las dimensiones del mapa y seguidamente separados por espacios los valores.

Formato salida: N n_0 n_1 … n_{N^2-1}

Comando: ‘pausar’

Formato: pausar

Descripción: Llama al método del moduloArduino encargado de pausar el arduino.

Formato salida: -

Comando: ‘reiniciar’

Formato: reiniciar

Descripción: Llama al método del moduloArduino encargado de reiniciar el arduino.

Formato salida: -

Comando: ‘nodosDeCamino’

Formato: nodosDeCamino

Descripción: Devuelve el VALOR DE CAMINO de todas las posiciones del mapa en un array continuo. Primero N identifica las dimensiones del mapa y seguidamente separados por espacios los valores.

Formato salida: N n_0 n_1 … n_{N^2-1}

Comando: ‘setmapa’

Formato: setmapa N n_0 n_1 … n_{N^2-1}

Descripción: Se crea el objeto de la clase Mapa para almacenar la información que llega. N identifica las dimensiones del mapa y seguidamente separado por espacios los VALORES DE CAMINO.

Formato salida: -

12.2 Protocolo de comunicación Arduino-Python

La comunicación entre Arduino y Python se realiza de forma asíncrona, se envía un comando a la otra parte y en un tiempo corto no definido se recibe si se requiere una respuesta. La conexión por el puerto serie se inicia cuando arranca el programa y se cierra cuando el programa termina. En este protocolo es necesario identificar el receptor del mensaje puesto que se producen llamadas en ambas direcciones.

Comando(Arduino): 'led'

Formato: led X

Descripción: Enciende o apaga el led 13 de la placa arduino según el valor de X, para X=1 se enciende, para valores diferentes se apaga.

Comando(Arduino): 'getstado'

Formato: getstado

Descripción: Petición de Python para recibir el estado actual de la placa arduino, en la solución final no se utiliza, pero fue útil durante la fase de desarrollo y pruebas.

Comando(Arduino): 'setmapa'

Formato: setmapa N n_0 n_1 … n_{N-1}

Descripción: Establece el array de ordenes con los VALORES DE DIRECCION para resolver el laberinto y cambia de estado a 'Recorriendo laberinto'. N identifica el número de órdenes que se van a recibir.

Comando(Arduino): 'reiniciar'

Formato: reiniciar

Descripción: Llama a la función reiniciar().

Comando(Arduino): ‘pausar’

Formato: pausar

Descripción: Si nos encontramos en el estado ‘Recorriendo laberinto’ guarda el tiempo restante para completar el movimiento actual y cambia el estado a ‘Pausado’, si ya se estaba en este estado se reestablece el tiempo restante y se vuelve al estado ‘Recorriendo laberinto’.

Comando(Python): ‘estado’

Formato: estado X

Descripción: Informa sobre el estado actual de la placa arduino. X identifica el VALOR DE ESTADO.

Comando(Python): ‘progreso’

Formato: progreso X

Descripción: Informa sobre el progreso del robot recorriendo el laberinto. X identifica el VALOR DE PROGRESO del robot.

13. CODIGOS DE REPRESENTACIÓN

En esta sección se exponen los códigos numéricos que se han utilizado para representar información no numérica de una forma más eficiente para la programación.

13.1 Mapa del laberinto (VALOR DE MAPA):

Cuando se construye un esquema de un laberinto es necesario identificar la casilla de inicio, de finalización y que casillas son transitables o no.

Estos valores se usan para definir qué contiene cada celda que compone el mapa del laberinto.

- El valor 0 identifica a la casilla de inicio, donde se comienza a recorrer el mapa.

- El valor 1 identifica las casillas que son transitables.

- El valor 2 identifica las casillas no transitables(paredes).

- El valor 3 identifica la casilla de finalización, donde se pretende llegar.

13.2 Estado del camino (VALOR DE CAMINO):

Cuando se resuelve un laberinto queda marcado sobre cada casilla (independientemente del VALOR DE MAPA de contenga) si el camino de la solución encontrada pasa o no sobre dicha casilla. Se han extendido los valores para indicar además si esa casilla ya ha sido recorrida.

- El valor 0 identifica una casilla donde no hay camino (ya sea la casilla transitable o no).

- El valor 1 identifica una casilla donde hay camino, pero aún no ha sido recorrido.

- El valor 2 identifica una casilla donde hay camino y ya ha sido recorrido.

13.3 Direcciones (VALOR DE DIRECCIONES):

Estos valores identifican valores enteros con direcciones:

- El valor 1 representa ‘Arriba’.

- El valor 2 representa ‘Derecha’

- El valor 3 representa ‘Izquierda’

- El valor 4 representa ‘Abajo’

13.4 Progreso (VALOR DE PROGRESO):

Estos valores identifican el progreso en el recorrido del laberinto, su valor oscila entre 0 y el número máximo de pasos desde el principio hasta el final del recorrido.

Por ejemplo, si un recorrido está formado por dos direcciones (Arriba, Derecha) el valor de progreso será 0 mientras aún no se esté desplazando, cuando se desplace en la primera dirección el progreso será 1 y finalmente 2 cuando alcance el final del laberinto.

13.5 Estado del dispositivo (VALOR DE ESTADO):

Estos valores identifican los posibles estados de la placa arduino. Son usados para modelar su máquina de estados:

-El valor 0 identifica al estado ‘Esperando mapa’

-El valor 1 identifica al estado ‘Recorriendo laberinto’

-El valor 2 identifica al estado ‘Pausado’

-El valor 3 identifica al estado ‘Terminado’

14. BIBLIOGRAFÍA

https://es.wikipedia.org (Información general)

http://php.net/manual/es/ (Manual de php)

http://www.w3schools.com/php/default.asp (Manual de php)

http://stackoverflow.com/ (Información general)

https://docs.python.org/ (Manual de Python)

https://www.arduino.cc/en/Reference/HomePage (Manual de Arduino)

https://msdn.microsoft.com/en-us/library/67ef8sbd(v=vs.140).aspx (Manual de C#)

http://www.w3schools.com/html/default.asp (Manual HTML)

http://www.w3schools.com/bootstrap/default.asp (Manual Bootstrap)